I claim:

1. A method of scheduling on one or more processors, executions of both periodic and asynchronous real-time processes having hard or soft deadlines, comprising automatically generating a pre-run-time schedule comprising mapping from a specified set of periodic process executions to a sequence of time slots on one or more processor time axes, each of the time slots having a beginning time and an end time, reserving each one of the time slots for execution of one of the periodic processes, a difference between the end time and the beginning time of each of the time slots being sufficiently long that execution of all of the periodic processes, including satisfaction of predetermined constraints and relations comprising at least one of release time, worst-case computation time, period, deadline, deadline nature, offset and permitted range of offset constraints, and precedence and exclusion relations and criticality levels can be completed between the beginning time and end time of respective time slots, and executing the processes in accordance with the schedule during run-time of the processor.

2. A method as defined in claim 1, including the step of converting at least one asynchronous process to a corresponding new periodic process prior to the mapping step, and mapping the new periodic process in a manner similar to mapping of other periodic processes.

3. A method as defined in claim 1, including the step of converting all asynchronous processes that can be contained without conflict in the time slots with periodic processes including new periodic processes converted from asynchronous processes, into new periodic

processes prior to the mapping step, and mapping all new periodic processes in a manner similar to mapping of other periodic processes.

4. A method as defined in claim 3, including further executing all non-converted asynchronous processes during run-time of the processor at times which do not interfere with execution of processes contained in the pre-run-time schedule.

5. A method as defined in claim 1 including, following pre-run-time scheduling and during run-time of the processor, the step of scheduling executions of a specified set of periodic and asynchronous processes such that all known ones of said specified constraints and relations will always be satisfied, the specified constraints and relations further comprising, for asynchronous processes, at least one of worst-case computation time, deadline, deadline nature and minimum time between two consecutive requests, and beginning time and end time of every time slot reserved for every periodic process execution in the pre-run-time schedule.

6. A method as defined in claim 3 including, following pre-run-time scheduling and during run-time of the processor, the step of scheduling executions of a specified set of periodic and asynchronous processes such that all known ones of said specified constraints and relations will always be satisfied, the specified constraints and relations further comprising, for asynchronous processes, at least one of worst-case computation time, deadline, deadline nature and minimum time between two consecutive requests, and beginning time and end time of every time slot reserved for every periodic process execution in the pre-run-time schedule.

7. A method as defined in claim 1, including scheduling, within the pre-run-time schedule, said difference between the end time and the beginning time of each of said periodic time slots with sufficient time capacity for execution of all asynchronous processes that have less latitude than considered ones of periodic processes in meeting their respective deadlines.

8. A method as defined in claim 3, including scheduling, within the pre-run-time schedule, said difference between the end time and the beginning time of each of said periodic time slots with sufficient time capacity for execution of all unconverted asynchronous processes that have less latitude than latitude than considered ones of periodic processes in meeting their respective deadlines.

9. A method for automatically adjusting lengths of periods of a predetermined set of periodic processes, comprising storing and sorting a list of reference periods, setting the length of the period of each periodic process to the length of the largest reference period that is no larger than an original period of the periodic process to form adjusted periods, and storing the adjusted periods for subsequent use in scheduling executions of the periodic processes.

10. A method as defined in claim 1, including prior to the mapping step, automatically adjusting lengths of periods of a predetermined set of periodic processes, by storing and sorting a list of reference periods, setting the length of the period of each periodic process to the length of the largest reference period that is no larger than an original period of the

periodic process to form adjusted periods, and storing the adjusted periods for subsequent use in pre-run-time scheduling of executions of the periodic processes.

11. A method as defined in claim 8, including prior to the mapping step, automatically adjusting lengths of periods of a predetermined set of periodic processes by storing and sorting a list of reference periods, setting the length of the period of each periodic process to the length of the largest reference period that is no larger than an original period of the periodic process to form adjusted periods, and storing the adjusted periods for subsequent use in pre-run-time scheduling of executions of the periodic processes.

12. A method as defined in claim 9, including setting the length of each reference period to be equal to the product of n powers, the base of each of the n powers being a distinct prime number, the n bases of the n powers being the first and smallest n prime numbers, the exponent of each of the n powers being bounded by an exponent upperbound, the sorted list of reference periods including every product of every said n powers.

13. A method as defined in claim 12, including controlling tradeoff between differences between original period lengths and the adjusted period lengths, and the length of a least common multiple of the adjusted period lengths, by values of the n exponent upperbounds and the value of n.

14. A method as defined in claim 1 including, prior to generating the pre-run-time schedule, determining whether each hard deadline asynchronous process should or should not be converted into a new

Exhibit 1003 - Page 0004

periodic process, and then converting a subset of a predetermined set of asynchronous processes having a worst-case computation time, minimum time between two requests characteristics and hard deadline constraints, which have been determined to be convertible, into a set of new periodic processes having release time, worst-case computation time, period, hard deadline, and permitted range of offset constraints, and reducing possible timing conflicts with other hard deadline periodic or hard deadline asynchronous processes with less latitude in meeting their deadlines, by taking into consideration the computation time requirements of the latter processes when determining the deadline of the new periodic process.

15. A method as defined in claim 1 including, prior to generating the pre-run-time schedule, determining whether each hard deadline asynchronous process should or should not be converted into a new periodic process, and then converting a subset of a predetermined set of asynchronous processes having a worst-case computation time, minimum time between two requests characteristics and hard deadline constraints which have been determined to be convertible, into a set of new periodic processes having release time, worst-case computation time, period, hard deadline, and permitted range of offset constraints, wherein a permitted range of offset of the new periodic process being is a subinterval of an interval or a full interval that begins at a predetermined start time, and ends at a time equal to the sum of the earliest time that an asynchronous process can make a request for execution plus the period length of the new periodic process minus one time unit.

16. A method as defined in claim 14, in which the determining step is performed by calculating whether a ratio of processing capacity of the processor which is required to be reserved for new periodic processes, to processor capacity that is required for the asynchronous process if left unconverted, exceeds a predetermined threshold value.

17. A method as defined in claim 1, including generating the pre-run-time schedule as a feasible two-part pre-run-time-schedule for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of all the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run-time schedule.

18. A method as defined in claim 1, including generating the pre-run-time schedule as a feasible two-part pre-run-time schedule on one or more processors for execution of periodic processes that may have non-zero

offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of all the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run-time schedule.

19. A method as defined in Claim 1, including generating the pre-run-time schedule as a feasible two-part pre-run-time schedule on one or more processors for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of all the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run-time schedule.

20. A method as defined in claim 17, including generating said pre-run-time schedule on a plurality of processors.

21. A method as defined in claim 14, including generating the pre-run-time schedule as a feasible two-part pre-run-time-schedule for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero length, and (b) a repeating part having length which is equal to a least common multiple of lengths of all the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that was converted from an asynchronous process, to generate said feasible pre-run-time schedule.

22. A method as defined in claim 21, including

generating said feasible pre-run-time schedule on one or more processors.

23. A method as defined in claim 21, further including the steps of generating the pre-run-schedule by constructing a schedule for all executions of the periodic processes within an interval starting from zero and having length equal to maximum offset value plus a bounded number of times of the length of a least common multiple of the periodic process periods, conditions for determining feasibility requiring the existence of a point in the pre-run-time schedule wherein starting from the latter point the schedule repeats in subschedule interval lengths equal to a least common multiple of lengths of all the periodic process periods, timing of all executions of all periodic processes within a time interval having length equal to the length of the least common multiple of the periodic process periods being included in each said repeating subschedule interval, and including satisfaction of all predetermined constraints and relations for all executions of all periodic processes within the subschedule interval starting from time zero and ending at said point plus the length of the least common multiple of the periodic process periods in the schedule.

24. A method as defined in claim 23, including generating said pre-run-time schedule on a plurality of processors.

25. A method as defined in claim 1, including generating the pre-run-time schedule on one or more processors as a feasible two-part pre-run-time-schedule for execution of periodic processes that may have non-zero offsets (a) an initial part which may be of zero

length, and (b) a repeating part having length which is equal to a least common multiple of lengths of all the periods of the periodic processes,

all executions of all periodic processes within a time interval of length equal to the length of the least common multiple of the periodic process periods being included in the repeating part of the pre-run-time schedule, wherein all said specified constraints and relations being satisfied for all executions of all periodic processes within both said initial part and said repeating part, and

using any offset value in a permitted range of offsets of each periodic process, including any offset value in the permitted range of offsets of any new periodic process that may have been converted from an asynchronous process, to generate said feasible pre-run-time schedule.

26. A method as defined in claim 1, further comprising the steps of:

(a) generating feasible said-run-time schedule for the execution of a set of hard deadline periodic processes, and of a set of soft deadline periodic processes,

(b) assigning a criticality level and a deadline upper-limit to each soft deadline periodic process,

(c) in the case of not finding a feasible pre-run-time schedule under said constraints and relations, identifying a soft critical set that contains soft deadline periodic processes for which modifying the deadlines of one or more processes in said soft critical set is necessary to meet the deadlines of all hard deadline processes,

(d) repeatedly selecting one process with a lowest

criticality level among processes for which its criticality level has not reached the deadline upper-limit in the soft critical set in each case in which a feasible pre-run-time schedule has not been found, increasing the deadline of the selected process by an amount that does not exceed the deadline upper-limit of the selected process and repeatedly attempting to find a feasible pre-run-time schedule until either a feasible pre-run-time schedule is found or all the deadline upper-limits of processes in the soft critical set have been reached without finding a feasible schedule, and indicating the event of either the inability of finding a feasible schedule or of finding a feasible pre-run-time schedule, and indicating the critical set when unable to find a feasible schedule,

(d) recomputing worst-case response times of all hard deadline asynchronous processes after a feasible pre-run-time schedule has been found for all hard and soft deadline periodic processes, and in every event that the worst-case response time exceeds the deadline of a hard deadline asynchronous process, repeatedly selecting one process that has a lowest criticality level among all soft deadline periodic processes that contribute to the worst-case response time of the asynchronous process and increasing the deadline of the selected soft deadline periodic process until either (i) the worst-case response time of every hard deadline asynchronous process is less than or equal to its deadline or (ii) all the deadline upper limits of the set of soft deadline periodic processes that contribute to the worst-case response time of some hard deadline asynchronous process that exceeds the deadline of said asynchronous process have been reached, and indicating the event of the latter events (i) or (ii) and the set.

27. A method as defined in claim 1, further comprising, during run-time:

(a) detecting, at any predetermined time, whether there exists a possibility that immediate execution of a particular hard deadline asynchronous process may cause the execution of any periodic process with less latitude in meeting a deadline of the latter periodic process as compared with latitude of meeting the deadline of the particular hard deadline asynchronous process which is to be delayed beyond a predetermined time limit, even if the periodic process is not ready for execution at said any predetermined time, and

(b) delaying execution of the hard deadline asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any time.

28. A method as defined in claim 1, further comprising, during run-time:

(a) detecting, at any predetermined time, whether there exists a possibility that immediate execution of a particular hard deadline asynchronous process may cause the execution of any periodic process with less latitude in meeting a deadline of the latter periodic process as compared with latitude of meeting the deadline of the particular hard deadline asynchronous process which is to be delayed beyond the end of the time slot of the periodic process in the pre-run-time schedule, even if the periodic process is not ready for execution at said any predetermined time, and

(b) delaying execution of the hard deadline asynchronous process if said possibility is found to

exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any time.

29. A method as defined in claim 1, further comprising, during run-time:

(a) detecting, at any predetermined time, whether there exists a possibility that immediate execution of a particular hard deadline asynchronous process may cause the execution of the asynchronous process, or the execution of some other asynchronous process, to extend beyond the beginning of the time slot of the periodic process in the pre-run-time schedule, even if the periodic process is not ready for execution of said any predetermined time, and

(b) delaying execution of the hard deadline asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any time.

30. A method as defined in claim 27, including carrying out the method on a plurality of processors.

31. A method as defined in claim 29, including carrying out the method on a plurality of processors.

32. A method as defined in claim 1, further comprising, during run-time:

(a) detecting, at any predetermined time, whether there exists a possibility that immediate execution of a

particular hard deadline asynchronous process may cause the execution of any periodic process to be delayed beyond the end of the time slot of that periodic process in the pre-run-time schedule, even if the periodic process is not ready for execution at said any predetermined time, and

(b) delaying execution of the hard deadline asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of said asynchronous process at said any time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any time.

33. A method as defined in claim 32, including carrying out the method on a plurality of processors.

34. A method as defined in claim 1, further comprising, during run-time:

(a) detecting, at any predetermined time, whether there exists a possibility that immediate execution of a particular first hard deadline asynchronous process may cause execution of any second hard deadline asynchronous process to be continuously blocked for a duration of the execution of the first asynchronous process and a duration of the execution of any periodic process, when the second asynchronous process has less latitude in meeting the deadline of the second asynchronous process as compared with the latitude of both the first asynchronous process in meeting the deadline of the first asynchronous process and the latitude of the periodic process in meeting the deadline of the periodic process, even if neither the periodic process nor the second asynchronous process are ready for execution at the predetermined time, and

(b) delaying execution of the first hard deadline asynchronous process if said possibility is found to exist, even if said possibility is the only reason for delaying the execution of the first asynchronous process at said any predetermined time, and even if the delay will cause the processor to be in an idle state for a time interval of non-zero length beginning from said any predetermined time.

35. A method as defined in claim 34 including carrying out the method on a plurality of processors.

36. A method as defined in claim 1 comprising during run-time, detecting at least one event of, at any point in time, whether some asynchronous process has arrived by said point in time, if some asynchronous process or periodic process has completed its computation at said point in time, and if said point in time is both the release time and beginning time of a time slot in the pre-run-time schedule for some periodic process, and activating a run-time scheduler at said point in time, and should said at least one event have occurred, determining whether each asynchronous process that has arrived but has not yet been completed should be delayed or immediately put into execution, and at least one of the further steps:

(a) delaying execution of a hard deadline first asynchronous process when the execution of some other hard deadline asynchronous or periodic process which excludes the first asynchronous process has already started by has not yet been completed,

(b) delaying execution of a hard deadline first asynchronous process when execution of some other hard deadline asynchronous or periodic process which has less or equal latitude compared with the latitude of the

first asynchronous process in meeting their respective deadlines has already started but has not yet been completed,

(c) delaying execution of a hard deadline first asynchronous process when execution of some other hard deadline first asynchronous process that excludes a periodic process with less or equal latitude as compared with the latitude of the first asynchronous process in meeting their respective deadlines has already started but has not yet been completed,

(d) delaying execution of a hard deadline asynchronous process in the event there exists the possibility that immediate execution of the latter hard deadline asynchronous process may cause execution of a first periodic process with less or equal latitude to be delayed, when execution of the first periodic process may be preempted by execution of some second periodic process, and the latter hard deadline asynchronous process cannot be preempted by the second periodic process,

(e) delaying execution of a hard deadline asynchronous process when it is not allowed to preempt execution of any first process that excludes some other second hard deadline asynchronous process which has latitude that is less than both said any first process and the latitude of said second asynchronous process, whereby blocking of the second asynchronous process by the duration of more than execution of processes with greater latitude thereof may be avoided,

(f) delaying execution of a hard deadline asynchronous process to allow preemption of execution of the asynchronous process by execution of a first periodic process that has latitude less than or equal to the latitude of the asynchronous process, when the asynchronous process does not exclude the first periodic

process and does not exclude any other asynchronous process with a latitude that is less than the latitude of the first periodic process, and there does not exist execution of some second periodic process that has not been completed such that execution of the second periodic process is ordered before execution of the first periodic process and execution of the time slot of the first periodic process is not nested within the time slot of the second periodic process in the pre-run-time schedule, and

(g) selecting, among all hard deadline asynchronous processes that have already arrived and are not yet completed and are not delayed, a process which has the least latitude for immediate execution, wherein if more than one process is selected, further selecting a process among them with a smallest index for immediate execution.

37. A method as defined in claim 1, comprising during run-time, detecting, in a case in which no hard deadline asynchronous process or periodic process that has started is to be immediately put into execution, conditions of whether there exists an execution of some first periodic process that is ready for execution and has not completed execution, and there does not exist any other execution of some second periodic process that has not yet completed, such that execution of the second periodic process is ordered before execution of the first periodic process in the pre-run-time schedule, and the time slot of the first periodic process is not nested within the time slot of the second periodic process in the pre-run-time schedule, and there does not exist any other execution of some third periodic process that is ready and has not completed execution, such that execution of the third periodic process is nested within

05360600.050796

the time slot of the first periodic process in the pre-run-time schedule, and beginning execution of the first periodic process immediately in the event said conditions are true.

38. A method as defined in claim 1, comprising during run-time, detecting the conditions in a case in which no hard deadline asynchronous process or periodic process that has started is to be immediately put into execution, whether there exists a soft deadline first asynchronous process with known characteristics that is ready and has not completed execution, and that there does not exist any other second process such that the second process excludes the first process or the first process excludes some third process such that the third process has a latitude that is less than the latitudes of both the first process and the second process, and execution of the second process has started but has not been completed, and in the event the conditions are true, selecting among all such first soft deadline asynchronous processes with known characteristics a process that has the shortest deadline, and in the event among such processes there are some processes that have already started, then selecting a process among them which has already started for immediate execution.

39. A method as defined in claim 1, comprising during run-time, detecting the conditions, in a case in which no hard deadline asynchronous process or periodic process that has started, or periodic process that is ready, or soft deadline asynchronous process with known characteristics that is ready for execution, is to be immediately put into execution, whether there exists a soft deadline first asynchronous process with unknown characteristics that is ready and has not completed

execution, and that there does not exist any other second process such that the second process excludes the first process or the first process excludes some third process such that the third process has a latitude that is less than the latitudes of both the first process and the second process, and the execution of the second process has started but has not been completed, and in the event the conditions are true then selecting among all such first soft deadline asynchronous processes with unknown characteristics, a process that has the shortest deadline, and if among such processes there are some that have already started, then selecting a process among them that has already started for immediate execution.

40. A method as defined in claim 1, including, for use in generating the pre-run-time schedule, determining a worst-case response time of each hard deadline asynchronous process that has not been converted into a periodic process using a formula comprising:

the sum of all worst-case computation times of all asynchronous processes and periodic processes that have less or equal latitude as compared with the latitude of the asynchronous in meeting their respective deadlines,

plus the maximum time that the asynchronous process may possibly be blocked by some asynchronous or periodic process that has greater latitude as compared with the latitude of the asynchronous process in meeting their respective deadlines,

plus the worst-case computation time of the asynchronous process multiplied by the number of periodic processes with which the asynchronous process has an exclusion relation.

41. A method as defined in claim 1, including, during a pre-run-time phase, determining by simulation a worst-case response time of each hard deadline asynchronous process corresponding to a feasible pre-run-time schedule of periodic processes consisting of an initial part of the pre-run-time schedule and a repeating part of the pre-run-time schedule, wherein for each point in time from zero to the end time of the repeating part of the run-time schedule minus one time unit, simulating execution of the asynchronous process using functions used to determine a run-time schedule and recording response time of the asynchronous process under the assumption that the asynchronous process starts at a point in time under consideration, all other asynchronous processes that can possibly block the asynchronous process arriving at one time unit prior to the point in time under consideration, and all asynchronous processes that have deadlines that are shorter than the deadline of the asynchronous process arriving at the same said point of time under consideration, scheduling executions of periodic processes to start at the beginning time and to complete execution at the end time of their respective time slots in the pre-run-time schedule, wherein whenever the asynchronous process is delayed because it may block execution of some periodic process having less latitude than the latitude of the asynchronous process, or may block execution of some second asynchronous process for the duration of more than one execution of processes having greater latitude as compared with the latitude of the second asynchronous process, all asynchronous processes having less latitude as compared with the latitude of the asynchronous process is delayed in order to delay the asynchronous process for a maximum possible

amount of time, thus simulating all possible worst-case scenarios of executions of the asynchronous process.

42. A method as defined in claim 1, including, during a pre-run-time phase, restricting every periodic process to be executed strictly within its time slot in the pre-run-time schedule by changing release time of execution of each periodic process to be equal to the beginning time of the time slot reserved for execution of the periodic process.

43. A method as defined in claim 1, including, during a pre-run-time phase, generating tables of safe start time intervals for the executions of hard deadline asynchronous processes, wherein every periodic process is scheduled to be executed strictly within its time slot in the pre-run-time schedule, by changing the release time of execution of each periodic process to be equal to the beginning time of the time slot reserved for execution of the periodic process, wherein for every point in time of the pre-run-time schedule, it is determined whether each asynchronous process should be delayed, under the assumption that the actual start time of execution of every periodic process is equal to the beginning time of its time slot, and the actual end time of execution of every periodic process is equal to the end time of its time slot, wherein for every point in time of the pre-run-time schedule, in the event the hard deadline asynchronous process is to be delayed according the assumptions, the point in time is set to be unsafe and recorded in a corresponding entry in the table for the point in time and the asynchronous process.

44. A method as defined in claim 1, including, during a pre-run-time phase, determining a worst-case

0960960.000140

response time of each soft deadline asynchronous process for which worst-case computation time, deadline and minimum time between two requests are known before run-time and that has not been converted into a periodic process, using a formula:

the sum of all worst-case computation times of all asynchronous processes that have less or equal latitude as compared with the latitude of the asynchronous process in meeting their respective deadlines,

plus the sum of all worst-case computation times of all periodic processes,

plus the maximum time that the asynchronous process may possibly be blocked by some asynchronous process that has greater latitude as compared with the latitude of the asynchronous process in meeting their respective deadlines.

45. A method as defined in claim 1, including, during a pre-run-time phase, determining by simulation a worst-case response time of each soft deadline asynchronous process for which the worst-case computation time, deadline, and minimum time between two requests are known before run-time, corresponding to a feasible pre-run-time schedule of periodic processes consisting of an initial part of the pre-run-time schedule and a repeating part of the pre-run-time schedule, wherein for each point in time from zero to an end time of the repeating part of the pre-run-time schedule minus one time unit, and simulating the execution of the asynchronous process using functions which are used in determining a run-time schedule and recording the response time of the asynchronous process under the assumption that the asynchronous process starts at the point in time and that all other soft

deadline asynchronous processes that can possibly block the asynchronous process arrive at one time unit prior to the point in time, and all soft deadline asynchronous processes that have deadlines which are shorter than the deadline of the asynchronous process and all hard deadline asynchronous processes arrive at the same point in time, all executions of periodic processes starting at the beginning time and completing at the end time of their respective time slots in the pre-run-time schedule, thus simulating all possible worst-case scenarios of executions of the asynchronous process.

46. A method as defined in claim 1, including, during a pre-run-time phase, generating tables of safe start time intervals for the executions of hard deadline asynchronous processes, wherein every periodic process is scheduled to be executed strictly within its time slot in the pre-run-time schedule, by changing the release time of execution of each periodic process to be equal to the beginning time of the time slot reserved for execution of the periodic process, wherein for selected points in time of the pre-run-time schedule, it is determined whether each asynchronous process should be delayed, under the assumption that the actual start time of execution of every periodic process is equal to the beginning time of its time slot, and the actual end time of execution of every periodic process is equal to the end time of its time slot, wherein for selected points in time of the pre-run-time schedule, in the event the hard deadline asynchronous process is to be delayed according the assumptions, that point in time is set to be unsafe and recorded in a corresponding entry in the table for the point in time and the asynchronous process.

47. A method as defined in claim 46, in which the selected points in time are every point in time.

48. A method as defined in claim 1, including, during a pre-run-time phase, determining by simulation a worst-case response time of each soft deadline asynchronous process for which the worst-case computation time, deadline, and minimum time between two requests are known before run-time, corresponding to a feasible pre-run-time schedule of periodic processes consisting of an initial part of the pre-run-time schedule and a repeating part of the pre-run-time schedule, wherein for selected points in time from zero to an end time of the repeating part of the pre-run-time schedule minus one time unit, simulating the execution of the asynchronous process using functions which are used in determining a run-time schedule and recording the response time of the asynchronous process under the assumption that the asynchronous process starts at the selected point in time and that all other soft deadline asynchronous processes that can possibly block the asynchronous process arrive at one time unit prior to the selected point in time, and all soft deadline asynchronous processes that have deadlines which are shorter than the deadline of the asynchronous process and all hard deadline asynchronous processes arrive at the same point in time, all executions of periodic processes starting at the beginning time and completing at the end time of their respective time slots in the pre-run-time schedule, thus simulating all possible worst-case scenarios of executions of the asynchronous process.

49. A method as defined in claim 48, in which the selected points in time are every point in time.

50. A method of processing a plurality of processes, some of which are periodic and some of which are asynchronous, comprising:

(i) prior to executing the processes on a processor:

(a) determining which asynchronous processes have less flexibility in meeting their deadlines than any of the periodic processes,

(b) adding the execution time of each of the less flexible asynchronous processes to the execution time of each of the periodic periodic processes,

(c) scheduling each of the periodic processes into time slots,

(ii) and during run-time of the processor:

(d) executing the periodic processes according to the schedule, interrupting any periodic process to execute an of said less flexible asynchronous processes for which a request to execute has been received by the processor,

(e) on receiving a request to execute an asynchronous process which has greater or equal flexibility in meeting their deadlines than any of the periodic processes, scheduling the requesting asynchronous process at a time which will not conflict with execution and completion of any of the periodic processes, and

(f) execute the scheduled latter asynchronous process at its scheduled time.

51. A method as defined in claim 50, including the step of converting at least one asynchronous process to a new periodic process prior to step (i)(a) in the event that a ratio of processor capacity which needs to be reserved for executing the new periodic process, to the processor capacity which needs to be reserved for

the asynchronous process if unconverted, exceeds a predetermined value, and the step of otherwise treating each new periodic process in a similar manner to other periodic processes.

52. A method of processing a plurality of processes, some of which are periodic and some of which are asynchronous, comprising:

(i) prior to executing the processes on a processor:

(a) determining which asynchronous processes have less flexibility in meeting their deadlines than any of the periodic processes,

(b) adding the execution time of each of the less flexible asynchronous processes to the execution time of each of the periodic processes,

(c) scheduling each of the periodic processes into time slots,

(d) converting each asynchronous process which has greater or equal flexibility in meeting their deadlines than any of the periodic processes into a new periodic process, and scheduling the new periodic process at a time which will not conflict with execution and completion of any of the other periodic processes,

(ii) and during run-time,

(e) executing the periodic and new periodic processes, interrupting any of the periodic and new periodic processes to processes any of the less flexible asynchronous processes for which a request to execute may be received at any time.

53. A method as defined in claim 52, including the step of converting at least one asynchronous process to a new periodic process prior to step (i)(a) in the event that a ratio of processor capacity which needs to be reserved for executing the new periodic process, to

the processor capacity which needs to be reserved for the asynchronous process if unconverted, exceeds a predetermined value, and the step of otherwise treating each new periodic process in a similar manner to other periodic processes.

54. A method as defined in claim 50, including adding the execution time of each of the less flexible asynchronous processes to the execution time of each of the periodic processes which has greater or equal flexibility in meeting their deadlines than any of the periodic processes, interrupting the scheduled latter asynchronous process to process any of the less flexible asynchronous processes for which a request to execute may be received at any time.

55. A method as defined in claim 51, including adding the execution time of each of the less flexible asynchronous processes to the execution time of each of the periodic processes which has greater or equal flexibility in meeting their deadlines than any of the periodic processes, interrupting the scheduled latter asynchronous process to process any of the less flexible asynchronous processes for which a request to execute may be received at any time.

56. A method of scheduling execution of processes by a processor comprising:

(a) scheduling periodic time slots for all periodic processes which time slots each include time for each of the periodic processes and time for all asynchronous processes which have less flexibility in meeting their deadlines than do the periodic processes,

(b) determining worst case response times of all other processes,

(c) construct a schedule which includes the periodic time slots and sufficient intervening time to process said all other processes, and

(d) execute the processes in accordance with the schedule and as said all other processes are required to be processed from time to time.

57. A method of scheduling execution of processes in a processor, wherein the processes may be periodic having at least one of

(a) hard deadlines with known characteristics (phk),

(b) soft deadlines with known characteristics (psk),

and may be asynchronous having at least one of

(c) hard deadlines (ahk), or

(d) soft deadlines (ask)

comprising:

A. in a first pre-run-time off-line phase:

I. dividing ahk processes into two subsets ahkp and ahka,

1. converting the ahkp processes into new periodic processes in the event that a ratio of processor capacity that needs to be reserved for the new periodic process, to the processor capacity that needs to be reserved for the ahkp process if unconverted, exceeds a predetermined threshold, and if not, designating the ahkp process as an ahka process,

2. retaining akha processes as asynchronous,

(i) reserving enough time in the new periodic processes deadline to accommodate times of all periodic and asynchronous processes that have less latitude in meeting their deadlines, thereby to allow said processes that have less latitude to preempt a new periodic process if possible at run-time,

(ii) reserving processor capacity for ahka processes by adding computation time of each ahka process to computation time of every periodic process that has greater flexibility in meeting its deadline than of the ahka process, thereby to allow each ahka process to preempt execution of any such periodic process which has the greater flexibility if possible at the run-time,

II.1. determining the schedulability of all phk processes having known characteristics and new periodic processes converted from ahkp processes,

2. constructing a pre-run-time schedule in which one or more time slots are reserved for execution of every phk process and every new periodic process converted from one ahkp process, wherein a time slot reserved for a phk process includes time reserved for execution of all ahka processes that have less flexibility in meeting their deadlines than the phk process,

3. adjusting the lengths of time slot periods by predetermined parameters for controlling balance between a utilization level and length of a pre-run-time schedule,

III.1. determining worst case response times of all ahka processes,

2. verifying schedulability of each ahka process by checking whether its deadline is greater or equal to its worst-case response time,

IV. 1. determining the schedulability of all psk processes, while maintaining the previously determined schedulability of all phk and ahka processes,

2. reconstructing a pre-run-time schedule in which one or more time slots are reserved for execution of every phk process including every new phk process converted from an ahka process, for every psk process,

Document ID: 5054305

and for execution of all ahka processes that have deadlines that are shorter than deadlines of the phk or psk process and thus may preempt execution of the phk or psk process,

3. determining, again, worst case response time of all ahka processes,

V. determining worst case response time of the ask processes,

VI. constructing a feasible pre-run-time schedule for all the periodic processes with known characteristics, and determining worst-case response time of all asynchronous processes with known characteristics,

B. and in a following run-time on-line phase,

I.1. scheduling execution of the periodic and asynchronous processes so as to guarantee that execution of every periodic process will be completed before ending of the time slot containing the periodic process in the feasible pre-run-time schedule, and to guarantee that execution of all asynchronous processes having soft deadlines and known characteristics is completed within the worst-case response time determined in step A(IV)(3) and in step V, and

2. scheduling all other processes on a best effort basis using remaining processor capacity.

Add A1

D1